Dynamic Extension of a Grammar-based Dialogue System: Constructing an All-Recipes Knowing Robot

Petra Gieselmann, Alex Waibel

Interactive System Labs,
Universität Karlsruhe, Am Fasanengarten 5
76131 Karlsruhe, Germany
petra@ira.uka.de, ahw@cs.cmu.edu

10.21437/Interspeech.2006-144

Abstract

In the upcoming field of humanoid and human-friendly robots, the ability of the robot for simple, unconstrained and natural communication with its users is of central importance. The basis for appropriate actions of the robot is the correct understanding of the user utterances. To be able to cover all the entities a user might talk about, we enhanced our dialogue manager with an ability for dynamic vocabulary generation out of information found across the internet. As a test case, we chose an internet recipe database integrated in the dialogue manager of our household robot so that it can understand several thousand recipes and ingredients now.
Index Terms: dialogue management, human-robot interaction, vocabulary extension

1. Introduction

Today, spoken dialogue systems are generally restricted to a fixed predefined grammar and vocabulary. This mandates that the lexicon and grammar must anticipate in advance all the entities a user might refer to. In addition, all new entities must be manually added to the system which is very time consuming. Therefore, this paper presents a dialogue system enhanced with a dynamic vocabulary capability: New structured information can be generated from a database connected to the internet and used during the human-robot conversation.

The underlying objective of this work is to build a dialogue system that can flexibly incorporate new information from dynamic information sources across the internet. The information from the internet is prepared and structured and finally stored in a database accessed by the dialogue manager at runtime. The new vocabulary is integrated in some grammar rules as terminals. As a test case, we choose an internet recipe database integrated in the dialogue manager of our household robot [1, 2] so that it is now able to understand several thousand recipes and ingredients. Therefore, an internet recipe database is consulted and recipe names, ingredients and the cooking methods found there are stored in a database used by our dialogue manager.

This paper deals with the dynamic extension of vocabularies used within a dialogue manager of a household robot. Section two gives an overview of related work concerning the integration of new concepts within dialogue managers. Section three deals with our dialogue manager. Mechanisms for inheritance and for vocabulary enhancements by means of a database are also explored. We explain the effects on the speech recognizer of the dynamic vocabulary extensions. Section four deals with experimental details and results. We explain details of the recipe application and preliminary user studies to evaluate it. Finally, section five gives a conclusion and an outlook on future work.

2. Related Work

Rayner and his colleagues use the well known plug and play concept and transfer it to the speech understanding domain within an intelligent room [3]: The system allows a dynamic reconfiguration of the language processing components during runtime so that new devices, such as lamps or beamers, can be added at every time. They have a core grammar which covers general user utterances and can be enhanced by adding a new device with new lexical entries or grammar rules. Their approach resembles to ours in so far as we also have a core grammar which is enhanced by new lexicon entries and grammar rules. The main difference lies in the fact that they only have predefined new lexicon entries resp. grammar rules which are also of a very limited number, whereas we don't know what we get back from the internet database. We do not use any predefined lexicon entries, but generate them based on the information found in the internet recipe database.

In addition, Chung et al. try to overcome the restriction of predetermined, fixed vocabulary. They present a spoken dialogue interface enhanced with a dynamic vocabulary capability in order to support context-specific vocabulary sets which can be changed at any time during the dialogue [4]. They also use dynamic information sources across the internet, as we do. As a test case, they choose the restaurant information domain and limit the dynamic generation to restaurant names. In so far, their approach is similar to ours, but we use an information-based dialogue management approach with an ontology to cover much more semantic information than their finite state approach. In addition, they use a general acoustic model for all the new words, whereas we apply different acoustic models for all new words which were automatically generated.

3. The Dialogue Manager

3.1. Tapas

We use the TAPAS dialogue tools collection [5] based on the approaches of the language and domain independent dialogue manager ARIADNE [6]. This dialogue manager is specifically tailored for rapid prototyping and features inheritance mechanisms within grammar and ontology development. We developed the domain- and language-dependent components, such as an ontology, a spec-

```
database Recipes obj_recipe
jpkg://localhost:5454/Recipe?jpkg {
    dbtable Recipe obj_recipe {
    dbfield name   = [generic:NAME];
    dbfield ing1 = [ING1];
    dbfield ing2 = [ING2];
    dbfield ing3 = [ING3];
    dbfield ing4 = [ING4];
  };
  };
```

Figure 1: Example of a Database Import Definition for Recipes

ification of the dialogue goals, a database, generation templates and a context-free grammar used by the speech recognizer and the dialogue manager at the same time.

The dialogue manager uses typed feature structures [7], to represent semantic input and discourse information. A context-free grammar enhanced by information from the ontology defining all the objects, tasks and properties about which the user can talk parses the user utterance. The parse tree is converted into a semantic representation and added to the current discourse. If all the information necessary to accomplish a goal is available in discourse, the dialogue system calls the corresponding service. Otherwise, the dialogue manager generates clarification questions to the user by means of generation templates.

3.2. Inheritance Mechanisms

We also have some kind of a core grammar, as mentioned by Rayner and his colleagues [3]. New parts of the grammar can use concepts from the core grammar and also inherit information from this core grammar by means of an ontology [8]. Therefore, the general domain-independent ontology consists of concepts such as different speech acts and general goals, objects and properties from which specific objects, actions and properties could then inherit in the domain-dependent part.

The dialogue manager's ontology defines actions, properties and object descriptions. They are used by the dialogue manager itself and for database import. For example, the node obj_recipe inherits from *generic:object* which is defined in the core ontology and has different instantiations loaded from the database. Another example is the node $obj_ingredient$ which inherits from $obj_eatable$ and $obj_eatable$ again inherits from *generic:object*.

In this way, $obj_eatable$ as well as $obj_ingredient$ are objects of the speech act to eat something or to bring something eatable to the user. Whereas only $obj_ingredient$ is an object of the speech act asking for a recipe including the named ingredients.

3.3. Enhanced Database Capabilities

Databases contain additional information for the objects, such as their state or place within the real world. Furthermore, by using database information it is possible to separate grammar development from database information which will only be known during runtime. One such example is a list of all the recipes and the necessary ingredients the user can ask questions about.

To understand spoken question such as "How can I cook R?", the recipe name R needs to be covered by the grammar. Therefore, during grammar development nonterminal symbols are defined with a database import statement. For example the nonterminal symbol obj_recipe is defined with an import statement that contains the database name, the table entry with its lexical representation, and its semantic representation:

```
<obj_recipe,N,_> = import jpkg://
localhost:5454/Env?jpkg Recipe
name {generic:NAME import};
```

In addition, the dialogue manager uses database information to disambiguate or extend user input. While object types are defined in the ontology, the objects themselves can be found in the database. If object types are found in the user input, the corresponding objects are requested from the database. Database definitions determine which object types trigger database requests, and how the database information is integrated into the discourse (cf. Figure 1).

For example, a user utterance such as "How can I cook Spaghetti Napoli?" is first analyzed and parsed by the grammar so that the goal finding a recipe can be selected. The object "Spaghetti Napoli" is defined in the grammar as referring to the object obj_recipe and can therefore be resolved via database access where all different recipe names can be found. In this way, the user utterance can be converted to a complete dialogue service, which is then executed by the system and the robot explains the user in detail how to cook Spaghetti Napoli. Again the database is consulted to get information on the different cooking steps.

3.4. Effects on the Speech Recognizer

Since the dialogue manager and the speech recognizer share the same grammar in order to avoid any inconsistencies in the linguistic resources, lots of new words for recipes and ingredients also appear now in the vocabulary of the speech recognizer. We used letter-to-phoneme rules to automatically get the phonetic transcription in the dictionary for all these new words [9] so that they can be recognized by the speech recognizer.

4. Experimental Details

4.1. Introduction

The recipe application is able to inform the user about recipes and their ingredients. The user can ask the robot for recipes by giving the name, such as "Robbi, please tell me how to cook Coque au Vin." or by giving some ingredients "What can I cook with tomatoes, peppers, and cucumbers?". In both cases, the dialogue manager searches its database for corresponding recipes. If more than one is found, the user gets a list of possible recipes or is asked to further specify the ingredients. Finally, if one recipe is selected, the user can ask the robot for a complete list of the ingredients and for the preparation steps necessary to cook it.

Since the cooking instructions sometimes get quite complicated and long so that the user cannot follow the spoken instructions easily, the robot can also display them on a screen to the user.

4.2. Recipe Database

We evaluated some recipe databases found across the internet and choose "http://fooddownunder.com/" because it contains the most recipes (More than 200 000 recipes). The recipes have been parsed and the information is stored in a mysql database. In this way, we can avoid online access to the internet database during runtime

which could be very time consuming. Furthermore, the application is independent and can also work properly, if the internet recipe site is down for one or another reason.

In addition, the user can also ask the robot for downloading more recipes from the internet, if the recipe he is searching for is currently not in the database. Then the database is accessed online and the information is written in the mysql database so that it is also available in the following dialogues. The new recipe names and ingredients are also added to the lexicon of the speech recognizer by means of letter-to-phoneme rules to assure that they can be recognized.

We cleaned the data we got from the internet because everybody can easily add new recipes to the recipe database. Since this results in strange recipe names, such as "Spaghetti Neopolitana From Dannii Minogue" or different recipes for the same recipe name, we excluded all recipes with more than four words and all duplicate recipe names. To avoid too long response times for the user, we randomly selected about 30 000 recipe names to be included in the database. However, the user can download new recipes any time he wants to.

We created a list with all the parts of recipe names excluding function words which allows the user not only to say the exact name of a recipe as it is stored in the database, but also a similar one consisting of all the content words. In this way, the query will also be successful, if the user only utters some parts of a recipe name, such as "Spaghetti Carbonara" instead of "Spaghetti alla Carbonara". In addition, a list of all the possible ingredients was created for searching the recipe name by ingredients. In this way, we are able to extract the ingredient name from the specified amount of this ingredient within the recipe.

4.2.1. Finding Recipes by Name or by Ingredients

To find a recipe by name, first the database is searched for the exact name as specified by the user. If no result can be found, a query to the database with parts of the recipe name excluding any function words is executed. The results are ranked according to the number of parts found in a database entry so that the recipe matching most of the words given in the user utterance is in the first place. The user gets a list of at most ten recipes back.

To find a recipe by ingredients, a similar procedure is used so that the result is again a ranked list of at most ten recipes according to the ingredients specified by the user. In this way, the user at least gets one recipe which at least covers one of the ingredients he specified. Furthermore, since the result is a ranked list, the robot mentions the most interesting recipes to the user at the very beginning.

Since it is a spoken dialogue, the number of recipes given to the user is always limited by the short term memory capacities of the user. Therefore, the robot only tells the user the first three recipes and he may ask for more recipes or narrows down his search by giving more ingredients. All together, the list of recipes includes at most ten recipes divided in subgroups of three recipes to avoid overloading the short term memory of the user.

As far as the user selects a recipe, this recipe stays in the discourse so that the user can refer to it also by pronouns or elliptical expressions while cooking. He can ask for the preparation method and also refer to the different steps to cook. In addition, the robot can give the user a list of all the missing ingredients.

Parsing Rate	90.0%
Av. # of Recipes Found	9.13
Av. # of Recipes Compatible to the User Utterance	3.71
Recall	0.74
Precision	0.40

Table 1: User Study: Parsing Rate, Average Number of Recipes Found and Average Number of Recipes compatible to the User Utterance per User, Recall and Precision of Recipes Found

4.3. User Study

4.4. Introduction

We made different user studies to evaluate the potential of the enhanced system in detail. We started with a small text-based user study with six users from different nationalities and asked them for their five favorite recipes in order to evaluate whether these recipe names are covered by the current application and which answers are given to the users. Another user test was performed with the whole system including a speech recognizer to see how the users get along with it and how they like it. Therefore, we asked other six users who were not familiar with the household robot to ask it for some recipes and evaluated the user satisfaction with a small post-test survey. We were specifically interested in the recognition rate and how this is influenced by the massive amount of automatically generated phonetic representations of recipe names and ingredients.

4.4.1. Text-based User Study

The results revealed that 90% of the user input can be parsed and understood by the system (See Table 1). Although none of the recipe names can be found in exactly the same way as the user specified it in the database, parts of the recipe names are available so that the user gets always some recipes back (on average 9 recipes).

However, the search with parts of the recipe name sometimes results in recipes which are very different from the one the user wanted. For example, if the user asks for "Rigatoni al Forno", he gets different "al forno " dishes, but none with "Rigatoni". This is due to the fact that the user query is divided into three different strings "Rigatoni + al + forno" which results in a higher ranking of all recipes consisting of the two strings "al + forno" than only one string "Rigatoni". This might be avoided by integrating a part-of-speech tagger so that the recipe name is tagged and only nominal parts are sent to the database. However, this is a tricky task because the recipe names are in different languages.

Furthermore, we manually evaluated how many recipes are compatible to the user intention. The average number of recipes compatible to the user utterance is only 3.7 per user (see Table 1). This means that although the user nearly always gets some recipes back from the system, these recipes will sometimes not suite his needs. This fact is also supported by the precision and recall values we evaluated, as they are used in the information retrieval community above all (cf. eg. [10]). Precision is a measure of the usefulness of the results; whereas recall is a measure of the completeness of the result list. Therefore, recall is defined as the number of relevant recipes in the result list divided by the number of relevant recipes in the whole collection. Whereas precision specifies how well the engine performs in not returning nonrelevant recipes and

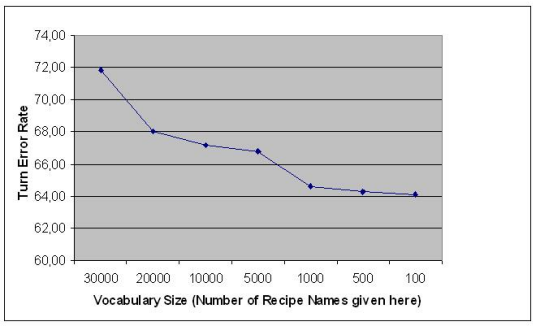

Figure 2: Turn Error Rates with Varying Vocabulary Sizes

is therefore defined as the number of relevant recipes in the result list divided by the total number of recipes in the result list. The recall is about 74% (cf. Table 1) which means that most of the relevant documents are retrieved. The only reason why the recall is below 100% is because we constrained the results presented to the user to ten at the most; otherwise all the recipes could have been found and the recall would have been 100%. Especially if the user asks for general recipes, such as "pizza" for example, he would get thousands of recipes. At the same time, the precision is quite low with about 40% so that the user also gets a lot of non-relevant documents back. This might be improved by sending more precise queries to the database and also carefully evaluating the results to exclude nonrelevant recipes. Within the speech-based user study, we also want to evaluate the user satisfaction and find out whether it is better to give the user more recipes or to give him less, but only the appropriate ones.

4.4.2. Speech-based User Study

Here, the users got six predefined tasks to accomplish by means of the robot starting from a very general one, where the users had to ask the robot for recipes in any way they want, to more specific ones, such as asking the robot for a specific recipe given the ingredients. After the test, the users answered some questions about their general impression of the system.

The results revealed that although the recognition accuracy was very low (28.19% of all the user utterances can be transferred to the correct semantics), the users managed to accomplish nearly all the tasks (91.67% task completion rate) and they had a generally positive attitude towards the robot (83.33% of the participants would use it to help them in the kitchen). The high turn error rate, ie. the rate of the user utterances which cannot be transformed to the correct semantics by the dialogue manager, was mostly due to the high perplexity of the recipe names. Namely, the 30 000 recipe names consisting again of different words can all be at the same place in a sentence according to the grammar. Therefore, we evaluated the effect of the perplexity with different vocbulary sizes ranging from 100 to 30 000 recipes: The results showed that the turn error rate decreases significantly with a decreasing vocabulary size from 71.81% to 64.09%, with an optimal vocabulary size of about 1 000 recipe names (cf. Figure 2).

In addition, the post-test survey showed that the users like to get many recipes back, although not all of them are relevant to their query. They prefer to get more recipes and not only the appropriate ones because they can browse these recipes in this way and find what they want.

5. Conclusion & Outlook

In this paper, we presented the concepts of dynamic vocabulary extension by means of information from the internet and how they can be used in dialogue management of human-robot communication. We explained the advantages of this concept which include the easy and fast generation of new information and the separation of information available at development time and runtime.

We took the example of a recipe database and evaluated whether the user can efficiently work with such an all-recipes knowing robot. Results are promising because the users liked to work with the robot, although the turn error rate was high because of the high perplexity. Further experiments revealed that the turn error rate can be decreased substantially by a decreasing vocabulary size with an optimal size of about 1 000 recipe names. Future experiments will also include other knowledge sources from the internet so that the robot gets much more background knowledge in general.

6. Acknowledgements

The authors would like to thank Landry Chouambe for programming parts of the recipe application and supporting the user experiments. This work was supported in part by the DFG within the SFB 588 on humanoid robots and by the EC under project CHIL (contract #506909).

7. References

[1] P. Gieselmann, C. Fügen, H. Holzapfel, T. Schaaf, and A. Waibel, "Towards multimodal communication with a household robot," *Proceedings of the Third IEEE International Conference on Humanoid Robots (Humanoids)*, 2003.

[2] R. Stiefelhagen, C. Fuegen, P. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel, "Natural human-robot interaction using speech, gaze and gestures," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[3] M. Rayner, I. Lewin, G. Gorrell, and J. Boye, "Plug and play speech understanding," *Proceedings of 2nd SIGdial Workshop on Discourse and Dialogue*, 2001.

[4] G. Chung, S. Seneff, C. Wang, and L. Hetherington, "A dynamic vocabulary spoken dialogue interface," *Proceedings of the Interspeech 04*, pp. 327–330, 2004.

[5] H. Holzapfel, "Towards development of multilingual spoken dialogue systems," *Proceedings of the 2nd LTC*, 2005.

[6] M. Denecke, "Rapid prototyping for spoken dialogue systems," *Proceedings of the 19th International Conference on Computational Linguistics*, 2002.

[7] B. Carpenter, *The Logic of Typed Feature Structures*, Cambridge University Press, 1992.

[8] M. Denecke, "Object-oriented techniques in grammar and ontology specification," *Proceedings of the Workshop on Multilingual Speech Communication*, 2000.

[9] A. Black, K. Lenzo, and V. Paget, "Issues in building general letter to sound rules," *Proceedings of the 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, 1998.

[10] W. S. Cooper, "On selecting a measure of retrieval effectiveness," *Jones, K. S. and Willett, P., (ed.): Readings in Information Retrieval*, 1997.